محمد امانلو

۸۱۰۱۰۰۰۸۴

تکلیف دستی دوم طراحی الگوریتم

استاد اسدپور

سوال ۲، صفحه ۱۳۶، تمرین ۴-۱ از روش decrease & conquer

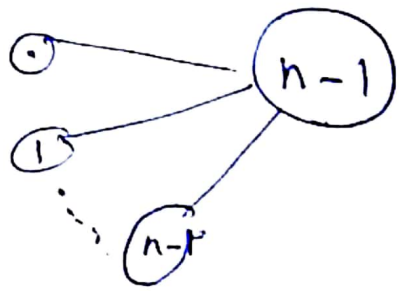
استفاده می‌کنیم، با شماره‌گذاری لیوان‌ها از ۱ تا ۲n در مجموع n جفت لیوان داریم که می‌خواهیم مرتب و منظم شود. هر مرحله جفت اول و جفت آخر را مرتب می‌کنیم یعنی لیوان ۲ را در لیوان ۱-۲n می‌ریزیم. در مرحله‌ی بعد با یه ۲-۸ جفت لیوان را مرتب کنیم که می‌شود ۲(n-۲)

لیوان، بنابراین در مجموع $\left\lceil \frac{n}{2} \right\rceil$ جا به جایی داریم. اگر n زوج باشد و اگر فرد $\frac{n-1}{2}$ جا به جایی داریم.

b. به طور کلی 2^n لیوان داریم. اگر لیوان‌ها را بخواهیم به صورت یکی در میان مرتب کنیم، می‌توانیم اینگونه تمامی لیوان‌های زوج پر باشند و همه‌ی لیوان‌های فرد خالی باشند. الان از لیوان اول شروع کرده و لیوان‌های فرد را بررسی می‌کنیم و به هر لیوان فردی که رسیدیم، آن را در اولین لیوان زوج خالی می‌ریزیم. به این ترتیب لیوان‌ها به صورت یکی در میان منظم و مرتب می‌شوند

سوال ۵، صفحه ۱۳۶، تمرین ۱-۴

غلط است. این گراف هم بند است ولی $n-1$ راس اول آن هیچ یالی به هم ندارند و اگر سوال را بیان $n-1$ راس اول حذف کنیم، گراف ناهمبند است ولی همه‌ی آن راس‌ها به راس n ام یال دارند پس گراف هم بند است. در خط $A[0 \dots n-2, 0 \dots n-2]$ return 0 if not connected



اگر راس‌های ۰ تا $n-2$ به هم وصل نبود، گراف ناهمبند است؟ (غلط بودم)

سوال ۹، صفحه ۱۳۶، تمرین ۱-۴ اگر لینک لیست ما، ۲ طرفه باشد اهمیتی ندارد و مرتبی نمی‌کند و الگوریتم دقیقا مشابه insertion sort است ولی در لینک لیست یک طرفه باید هنگام جای‌گذاری عنصر a در لیست، لیست مرتب شده‌ی خود را از چپ به راست بررسی می‌کنیم و با a مقایسه کنیم. پیچیدگی دقیقا مشابه insertion sort از n^2 order است

سوال ۱۲، صفحه ۱۳۶. اگر step_size ۳ و ۱ و ۴ باشد

۹)

a. SHELL SORTISUSEFUL

↕ ↕

SHEL

EFELi SORT ISUSSHUL

ISUS

SORT

FELL

E

S S U T

S O U S

I H R L

F E L L

E

EFLLIHRLSOUSSSUT

insertion sort

→ EEFHILLLORSSSTUV

b. سوال ۱۲، صفحه ۱۳۶ خیر stable نیست بسته به step-size مثلن است ترتیب ۲ مقدار برابر عوض شود. برای مثال ۵ و ۳, ۴ و ۳, ۲ و ۵, ۶

isteksize = 4:

۲ ۳ ۴ ۳
۶ ۵
۲ ۳ ۶ ۵
۴ ۳

⟹ ۴ ۳ ۲ ۳ ۶ ۵

عدد ۳ که در انتها قرار داشت، بعد از مرحله اول قبل از ۳ اولی قرار گرفت

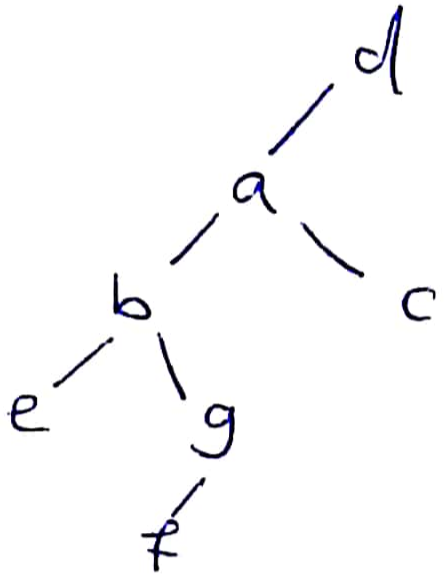
سوال ۱ ، صفحه ۱۴۲ ، تمرین ۲-۴

a. می‌خواهیم d را با DFS ریشه‌ی درخت

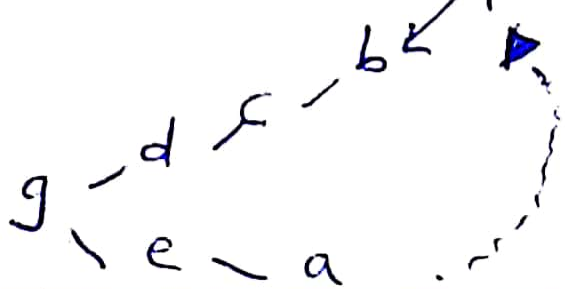
ترتیب خروج راس‌ها از استک

e f g b c a d

↳ topological sorting order: (d a c b g f e))



b. هنگام کشیدن درخت DFS یک back_edge دارد پس گراف جهت‌دار یک dag نیست و دارای دور است.



سوال ۲ . صفحه ۱۴۸ . تمرین ۴-۳

a. 1

insert 2 12,21

insert 3→12 312,132,123

insert 3→21 321,231,213

insert 4 → 312 4312,3412,3142,3124

insert 4 → 132 4132,1432,1342,1324

insert 4 → 123 4123,1423,1243,1234

insert 4 → 321 4321,3421,3241,3214

insert 4 → 231 4231,2431,2341,2314

insert 4 → 213 4213,2413,2143,2134

b. 12 3 4 , 1 2 43 , 1 4 2 3 , 4 1 2 3 , 4 1 2 3 , 4 1 3 2

1 4 3 2 , 1 3 4 2 , 1 3 2 4 , 3 1 2 4 , 3 1 4 2

3 4 1 2 , 4 3 1 2 , 4 3 2 1 , 3 4 2 1 , 3 2 4 1

3 2 1 3 , 2 3 1 4 , 2 3 4 1 , 2 4 3 1 , 4 2 3 1

4 2 1 3 , 2 4 1 3 , 2 1 4 3 , 2 1 3 4

c. 1234, 1243, 1324, 1342, 1423, 1432, 2134, 2143

2314, 2341, 2413, 2431, 3124, 3142, 3214, 3241

3412, 3421, 4123, 4132, 4213, 4231, 4312, 4321

سوال ۴، صفحه ۱۴۱، تمرین ۴.۳

a. for n=2

12, 21

for n=3 123, 213

312, 132

231, 321

for n=4 1234 2134 3124 1324 2314 3214

4231 2431 3421 4321 2341 3241

4132 1432 3412 4312 1342 3142

4123 1423 2413 4213 1243 2143

b.

$c(n)$ = تعداد بارهایی که الگوریتم جایگشت تولید میکند

$$c(n) = \sum_{i=1}^{n} c(n-1) \quad \text{یا} \quad c(n) = n^{c(n-1)} \quad \text{for } n>1,\ c(1)=1$$

یا سخن برابر است با $c(n)=n!$

c. $s(n) = \sum_{i=1}^{n} (s(n-1)+1)$ or $s(n) = n\,s(n-1) + n$

for $n>1$, $S(1) = 0$ طرفین را بر $n!$ تقسیم میکنیم $\frac{S(n)}{n!} = \frac{S(n-1)}{(n-1)!} +$

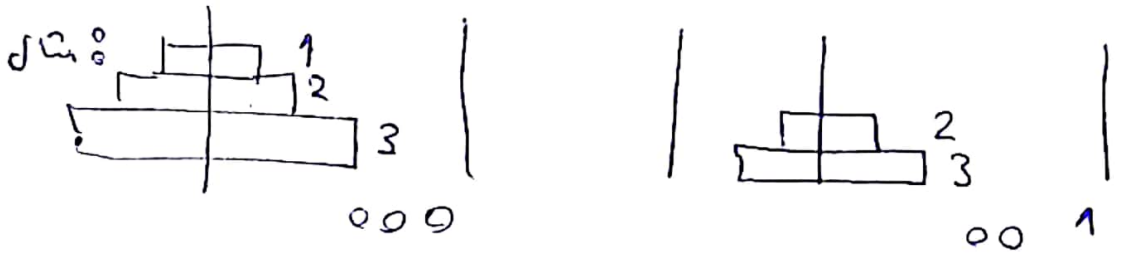
$\frac{1}{(n-1)!}$ $n>1$, $S(1) = 0$ $T(n) = T(n-1) + \frac{1}{(n-1)!}$ $n>1, T(1)=$

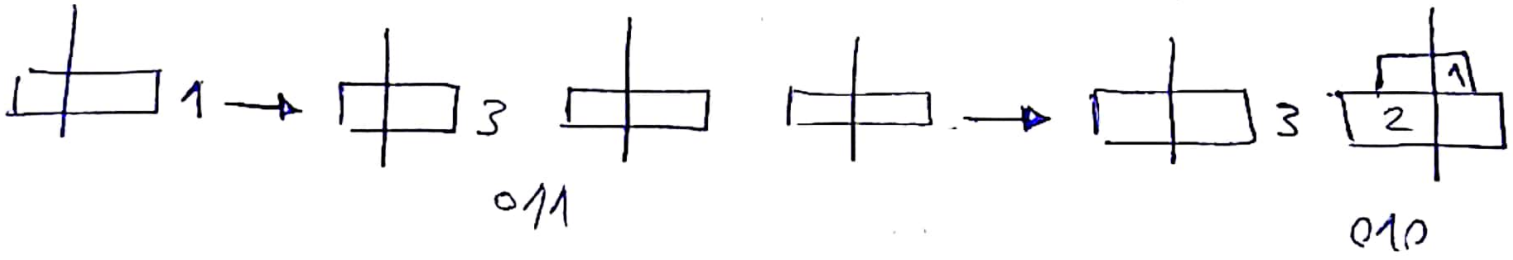
$$T(n) = T(1) + \sum_{i=1}^{n-1} \frac{1}{i!} = \sum_{i=1}^{n-1} \frac{1}{i!}$$

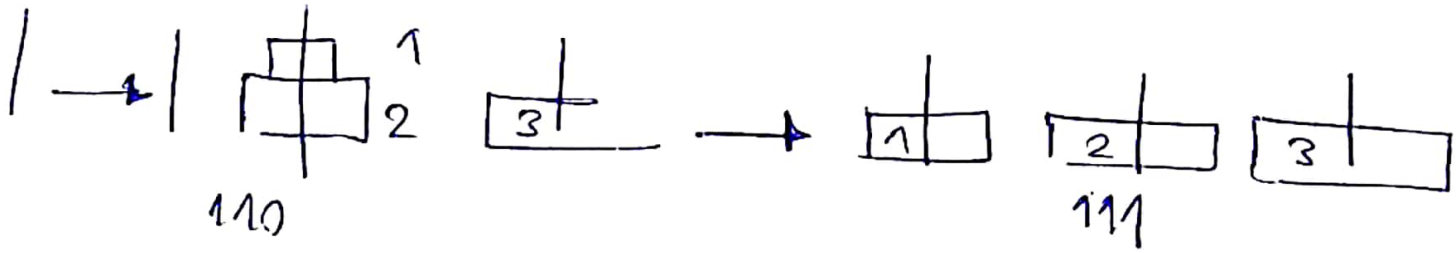
$$S(n) = n! \sum_{i=1}^{n-1} \frac{1}{i!} \approx n! \left(e - 1 - \frac{1}{n!}\right) \in \Theta(n!)$$

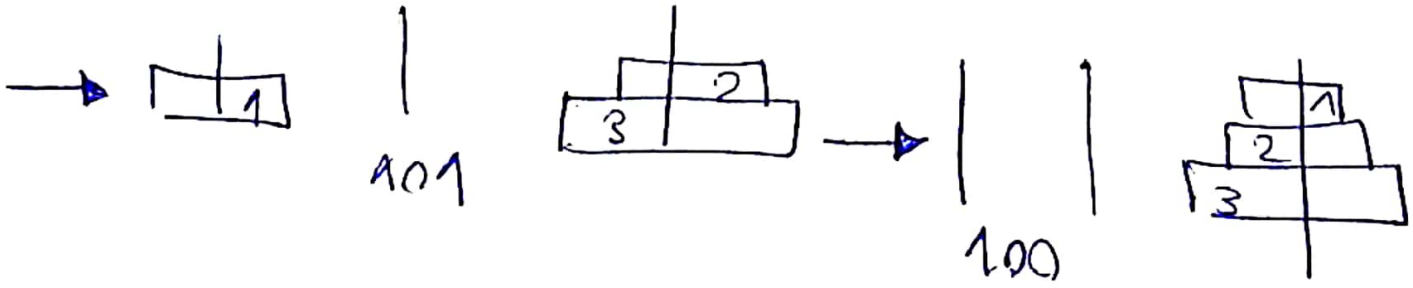
سوال ۱۱. صفحه ۱۴۸: تمرین ۴۳ ابتدا دیسک‌ها را از ۱ تا n شماره‌گذاری می‌کنیم. ۱ کوچک‌ترین دیسک و n بزرگ‌ترین دیسک. این n دیسک نشان‌دهنده‌ی n بیت ما هستند. دیسک کوچک راست‌ترین بیت (کم ارزش‌ترین) دیسک بزرگ چپ‌ترین (پرارزش‌ترین) ما باشد. با حرکت هر دیسک بیت مربوط به آن دیسک تغییر می‌کند. (از ۰ به ۱ یا از ۱ به ۰) در ابتدا همه‌ی بیت‌ها صفر می‌باشند و هیچ حرکتی نداشته‌ایم. با حرکت دادن دیسک i بیت i‌ام از راست تغییر می‌کند.

مثال:









b. کاملاً برعکس بالا باید عمل کرد. در هر مرحله یک بیت از عدد binary تغییر می‌کند. جایگاه آن بیت نشان‌دهنده شماره دیسکی است که باید جابه‌جا شود. دیسک‌ها را از کوچک به بزرگ از ۱ تا n شماره می‌گذاریم. عدد binary

هم n بیتی است . کم ارزش‌ترین بیت مربوط به کوچک‌ترین دلسگ و پرارزش‌ترین بیت مربوط به بزرگ‌ترین بیت است . تغییر هر بیت تنها شماره‌ی دلسگی که باید جابه‌جا بشود را برای ما مشخص می‌کند . بجل قرار دادن دلسگ کجاست از این قاعده استفاده می‌کنیم . دلسگ های فرد را روی دلسگ های زوج بزرگ‌تر از خود می‌گذاریم و دلسگ های عزد را روی دلسگ های زوج بزرگ‌تر از خود در هر مورد اگر چنین دلسگی وجود نداشت ، دلسگ را در یک میله‌ی خالی قرار می‌دهیم

سوال ۱۱ . صفحه ۱۵۶ . تمرین ۴-۴

a. n	m	
26	47	
13	94	
6	188	+ 94
3	376	
1	752	+ 376
	752	+ (376+94) = 1222

b . در الگوریتم دهقان روسی اگر n.m حساب کنیم ، $\log_2^n$ تا تکرار داریم اگر m.n را حساب کنیم ، $\log_2^m$ تا تکرار داریم .

اگر n کوچک‌تر باشد ، حالت اول سریع‌تر است و اگر m کوچک‌تر باشد ، حالت دوم سریع‌تر است

سوال ۵. صفحه ۱۶۷. تمرین ۵-۴. به طور کلی interpotation search

هنگامی درست عمل می‌کند که لیست توزیع یکنواختی داشته باشد. اما وزنی

که به برنامه می‌دهیم توزیع یکنواختی نداشته باشد، و عنصری لیستی که به دنبال آن هستیم

کنار آخرین یا اولین عنصر باشد توزیع یکنواختی عملکرد خوبی ندارد.

و 10^{0} × ۸ و ۱۰۰ و ۰۰۰ و ۳ و (۲) و ۱

مورد آخر توزیع نرمال را برهم می‌زند

لیست شده

سوال ۱۳ . جستجو را همیشه از اولین ردیف و آخرین ستون آن شروع می‌کنیم،
اگر عنصر کلید ما با آن برابر بود و جستجو تمام است ولی اگر عنصر از آن بزرگتر
بود ، قطعاً کلید دیگر در آن ردیف نمی‌تواند باشد ، می‌توان همه‌ی عناصر آن ردیف
را از جستجو حذف کرد . اگر عنصر از آن کوچکتر بود به طریق مشابه کلید دیگر
در عناصر آن ستون نمی‌تواند یافت شود پس آن ستون را هم می‌توان از جستجو

حذف کرد. به همین ترتیب هر حمله یک سطر یا یک ستون از جستجو حذف می‌شود و چون تعداد عنصرهای حذف‌شده در هر سطح متفاوت است، به روش table_size_decrease مسئله را حل کردیم. در هر حرکت فقط یک سطر یا یک ستون حذف می‌شود. پیچیدگی زمانی الگوریتم خطی بوده و از order می‌باشد. $O(n)$ ← چون n سطر و ستون داریم در کل.